



Umbraco Community Day 2023

Going Headless with MVC

Roy Berris





What will I be talking about?

Software Design

Roy Berris

Software Engineer at iO





Dissecting

Headless

*A software application or program
that functions without a front-end*

Let's decapitate a website

What are we left with?

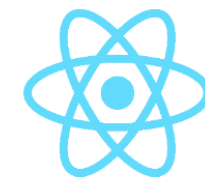
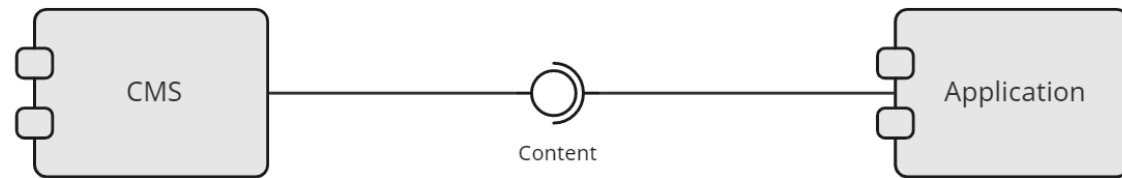
- **Data**



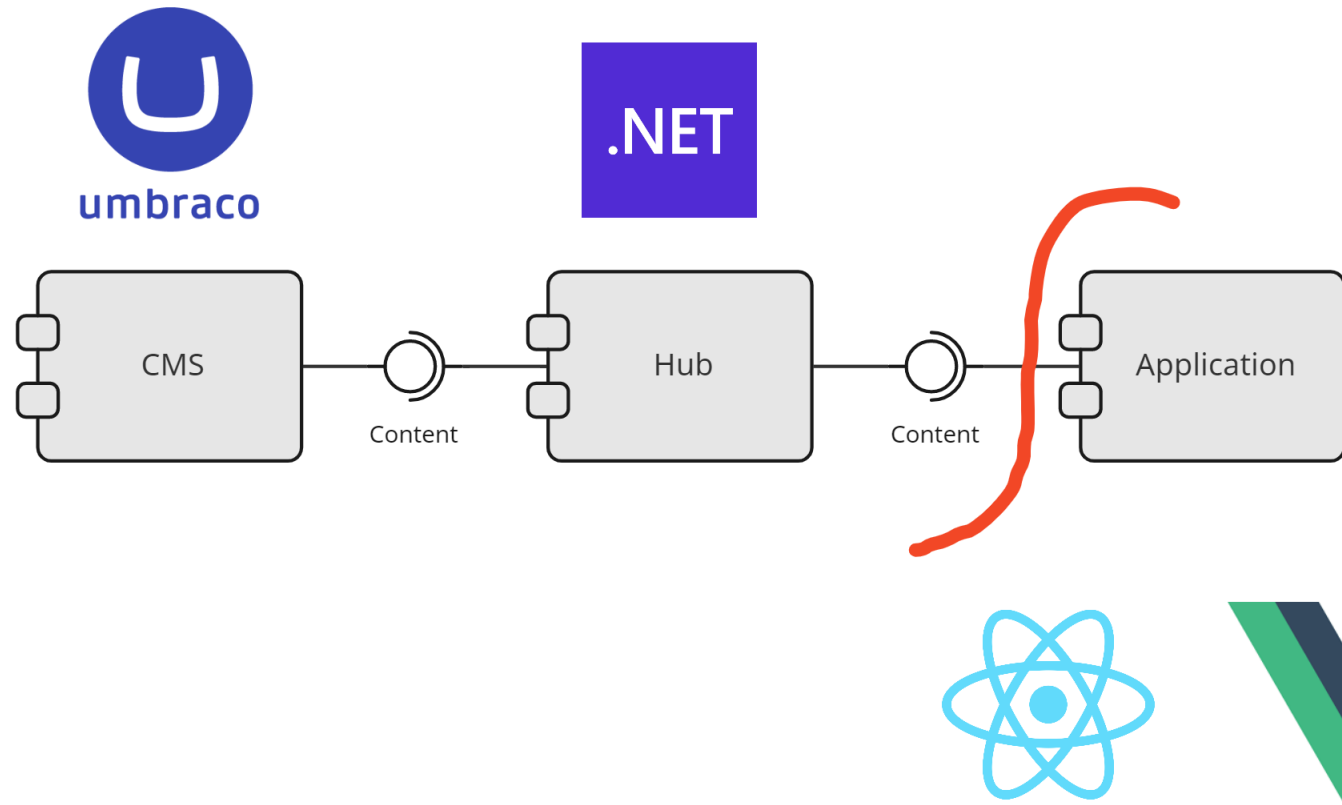
Looking at different

Components

Headless



Content Hub





Decoupling data

Content vs Content

From a design perspective



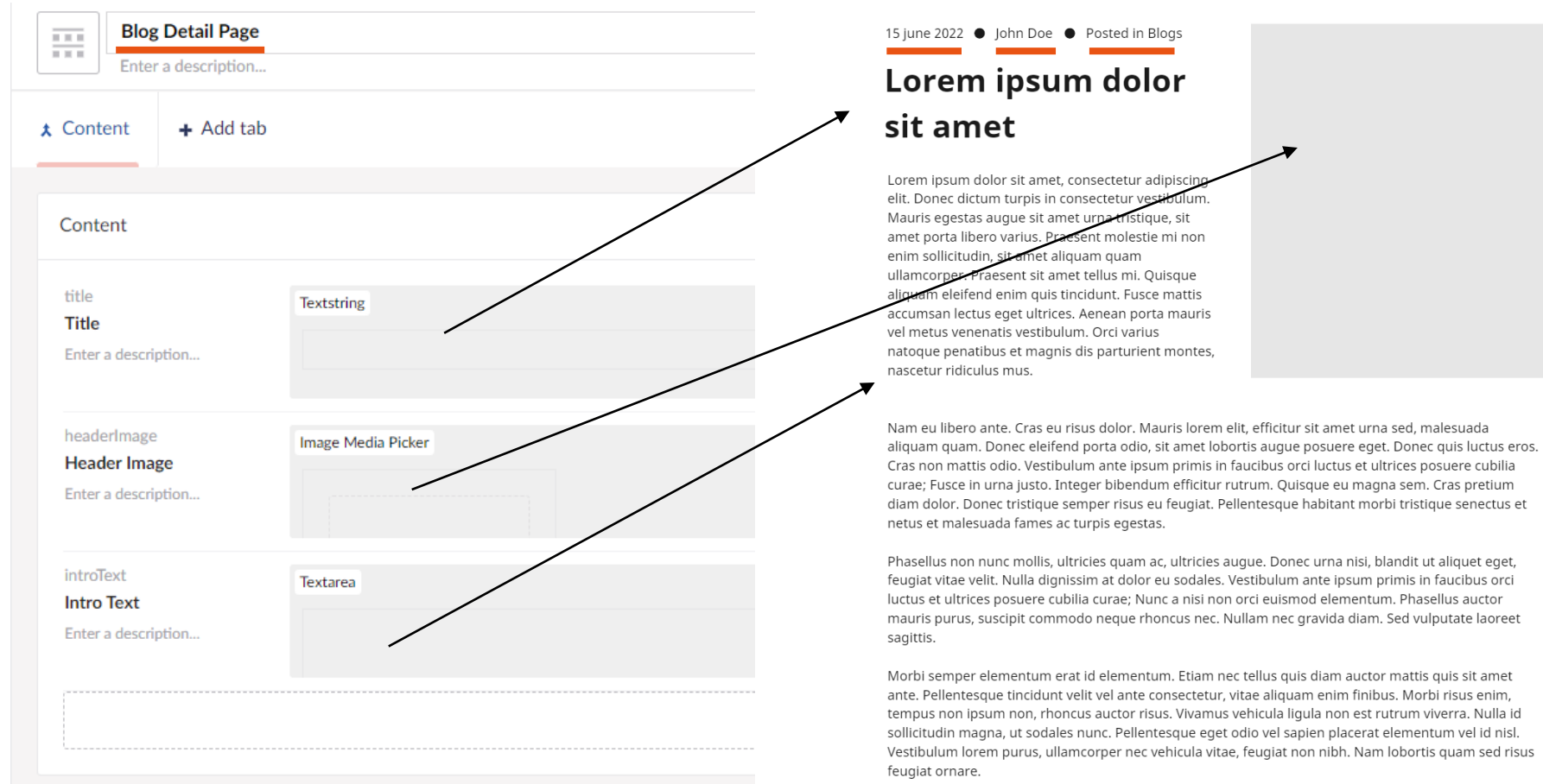
CMS Content

- Optimized for content editor
- Scattered
- Original

Application Content

- Optimized for consumption
- Normalized
- Derived

Example





Why do we want

Decoupling

It's all about flexibility

- Omni-channel
- Reusability
- Mocking

Two views

- **A headless CMS has a decoupled architecture**
- **Not every decoupled CMS is headless**



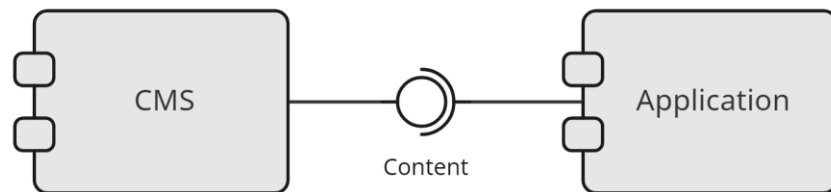


What are

Boundaries

A conceptual line

- Different interests
- Different actors
- Does this really belong here?





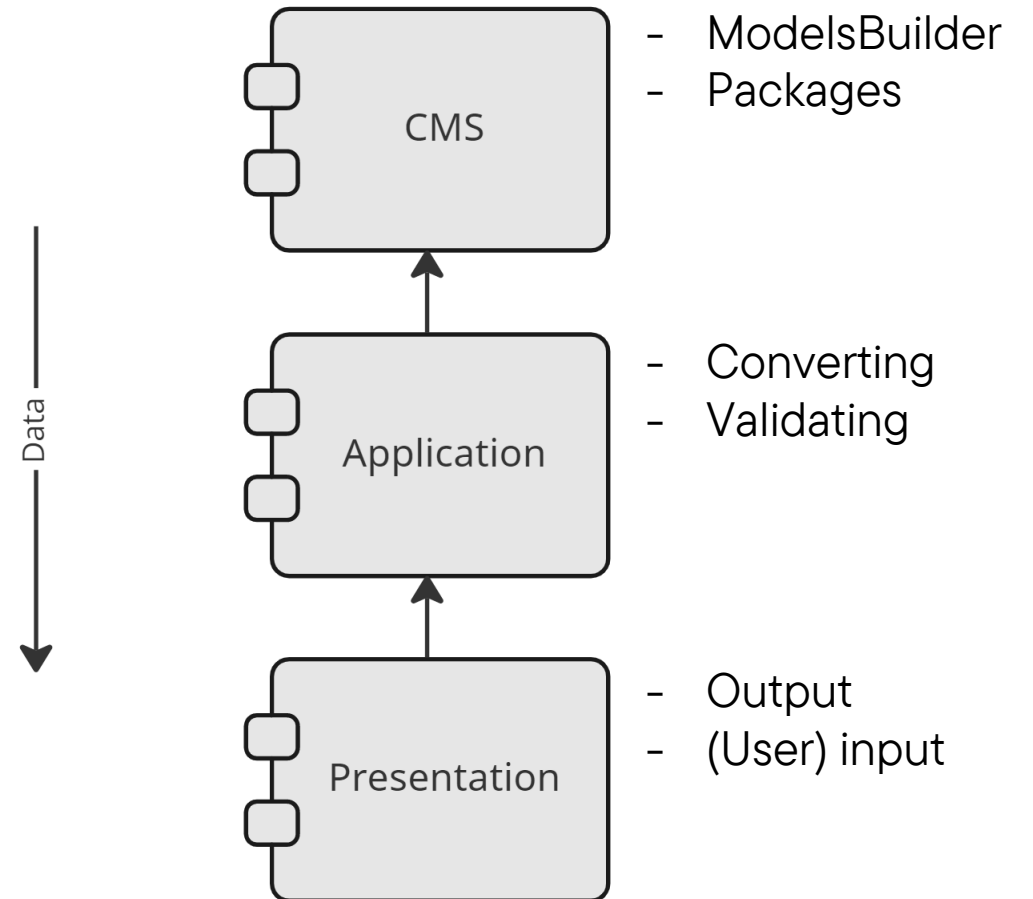
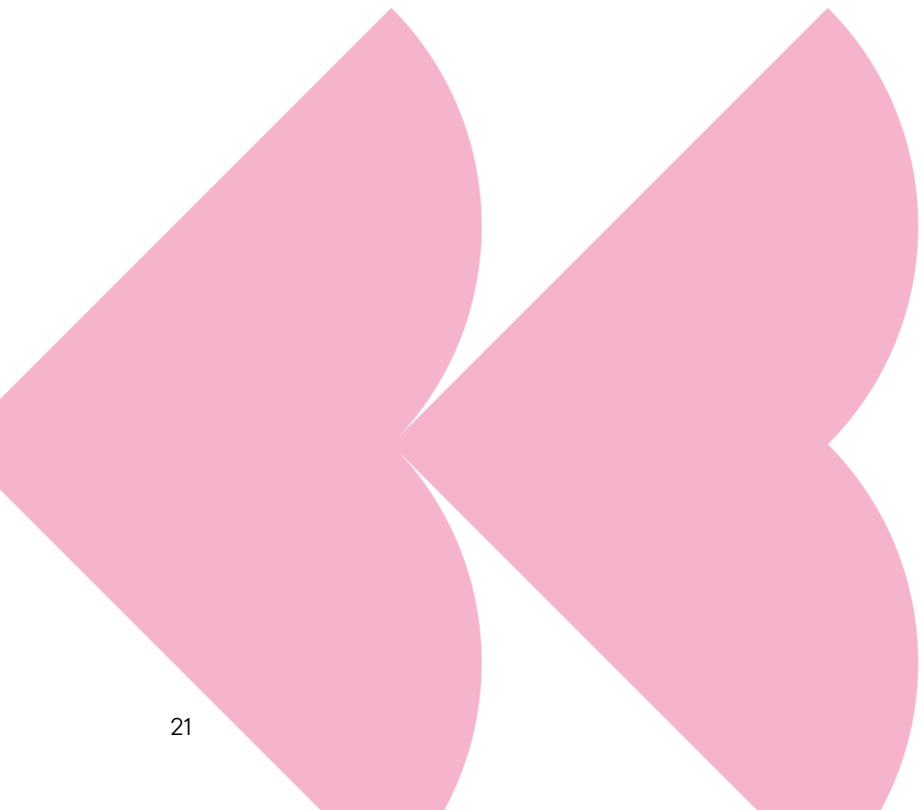
Decoupling and Umbraco



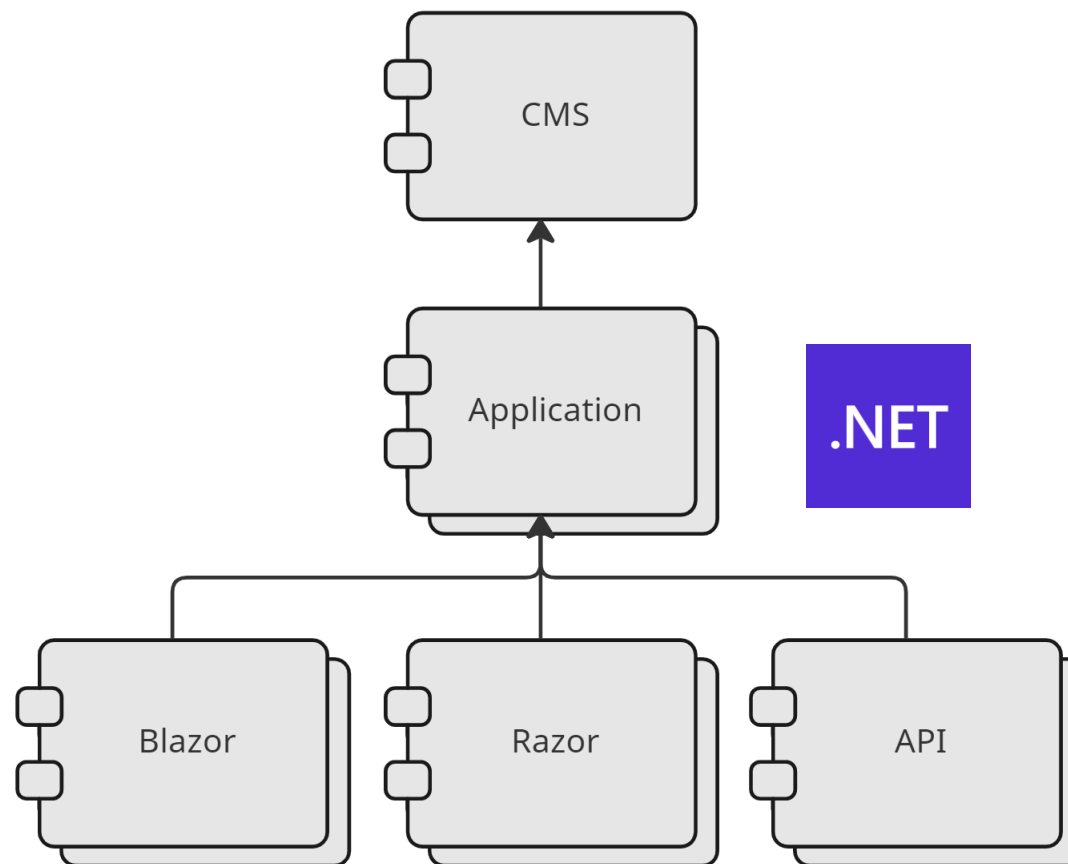
Looking at the

Project Set-up

3 layers



Stackable layers






Ah, good ol' Umbraco

CMS

All of Umbraco

- ModelsBuilder
- CMS Customization
- Anything CMS related

A document type



Blog Page
Enter a description...

Page

+ Add tab

Content

introText

Intro Text

Enter a description...

Textarea

title

Title

Enter a description...

Textstring

ModelsBuilder

```
"ModelsBuilder": {  
  "AcceptUnsafeModelsDirectory": true,  
  "ModelsDirectory": "~/../CMS/Models",  
  "ModelsMode": "SourceCodeManual",  
  "ModelsNamespace": "CMS.Models"  
}
```



What to do with

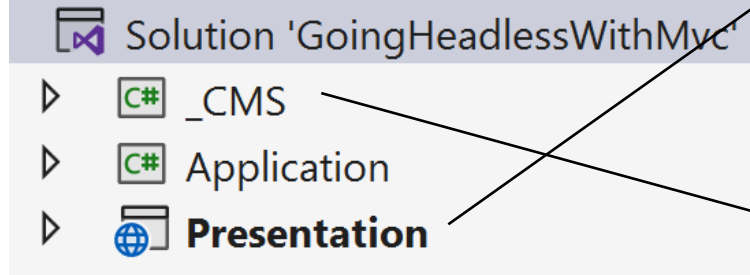
Presentation

Basically, no Umbraco

- No ModelsBuilder models
- No UmbracoViewPage
- High-level routing

Treat .NET as a coincidence

The exception



Startup.cs

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddUmbraco(_env, _config)
        .AddBackOffice()
        .AddWebsite()
        .AddComposers()
        .Build();

    services.AddCmsServices(_config, _env);
    services.AddApplicationServices();
    services.AddPresentationServices();
}
```

ConfigureServices.cs

```
public static class ConfigureServices
{
    public static IServiceCollection AddCmsServices(
        this IServiceCollection services,
        IConfiguration configuration,
        IHostEnvironment environment)
    {
        if (environment.IsEnvironment("test"))
        {
            // Do something
        }

        return services;
    }
}
```

A controller

```
public class DefaultRenderController : RenderController
{
    private readonly IContentResolver _contentResolver;

    public DefaultRenderController(ILogger<RenderController> logger, ICompositeViewEngine
        : base(logger, compositeViewEngine, umbracoContextAccessor)
    {
        _contentResolver = contentResolver;
    }

    public override IActionResult Index()
    {
        var publishedContent = UmbracoContext.PublishedRequest?.PublishedContent;

        if (publishedContent is null)
        {
            return NotFound();
        }

        var result = _contentResolver.Resolve(publishedContent);

        return View(result.Template, result.ContentModel);
    }
}
```

Presentation -> ConfigureServices.cs

```
services.Configure<UmbracoRenderingDefaultsOptions>(c =>
{
    c.DefaultControllerType = typeof(DefaultRenderController);
});
```



So, what's up with

Application

More Umbraco, no ASP.NET

- House of Models
- Application Logic
- Converting

Wrapping Data

```
public interface IContentResolver
{
    public ContentWrapper Resolve(IPublishedContent content);
}
```

```
public class ContentWrapper
{
    required public IContentModel ContentModel { get; set; }

    required public Guid Key { get; set; }

    required public string Template { get; set; }

    required public string Url { get; set; }
}
```

Converting Content

- Mapping 1 on 1
- Mapping to interfaces
- Real scenario will require more

Mapping Strategy

- Register to DI
- 1 implementation per published element

```
public interface IMappingStrategy<T>
    where T : IPublishedElement
{
    public IContentModel Map(T content);
}
```

News Page

```
public class NewsStrategy : IMappingStrategy<NewsPage>
{
    private readonly ICultureDictionary _dictionary;

    public NewsStrategy(ICultureDictionaryFactory dictionaryFactory)
    {
        _dictionary = dictionaryFactory.CreateDictionary();
    }

    public IContentModel Map(NewsPage content)
    {
        return new ContentDetailPage
        {
            Title = content.Title ?? string.Empty,
            IntroText = content.HeaderIntroText,
            Category = _dictionary["Category.News"] ?? "Nieuws",
            PublishDate = content.UpdateDate
        };
    }
}
```

Content Resolver

- Create the content model from a published element
- Retrieve the used template
- Add the content key

```
public class ContentResolver : IContentResolver
{
    private readonly IServiceProvider _serviceProvider;
    private readonly IFileService _fileService;
    private readonly IPublishedUrlProvider _publishedUrlProvider;

    public ContentResolver(
        IServiceProvider serviceProvider,
        IFileService fileService,
        IPublishedUrlProvider publishedUrlProvider)
    {
        _serviceProvider = serviceProvider;
        _fileService = fileService;
        _publishedUrlProvider = publishedUrlProvider;
    }

    public ContentWrapper Resolve(IPublishedContent content)
    {
        var contentModel = ResolveContentModel(content);
        var template = content.GetTemplateAlias(_fileService);

        return new ContentWrapper
        {
            ContentModel = contentModel,
            Key = content.Key,
            Template = template,
            Url = _publishedUrlProvider.GetUrl(content, mode: UrlMode.Absolute)
        };
    }
}
```



Content Resolver

- Create the content model from a published element
- Retrieve the used template
- Add the content key

```
private IContentModel ResolveContentModel(object content)
{
    if (content is not IPublishedElement)
    {
        throw new ArgumentException(
            $"Cannot parse content to {nameof(IPublishedElement)}");
    }

    var contentType = content.GetType();
    var strategyInterfaceType = typeof(IMappingStrategy<>).MakeGenericType(contentType);

    var strategy = _serviceProvider.GetRequiredService(strategyInterfaceType);

    if (strategy is null)
    {
        throw new InvalidOperationException(
            $"Cannot get mapping strategy for content type {contentType.Name}");
    }

    if (strategy?.GetType()?
        .GetMethod(nameof(IMappingStrategy<IPublishedElement>.Map))?
        .Invoke(strategy, new[] { content }) is not IContentModel executedMethod)
    {
        throw new InvalidOperationException(
            $"Cannot execute mapping strategy for content type {contentType.Name}");
    }

    return executedMethod;
}
```





Yeah, yeah, get to the point already

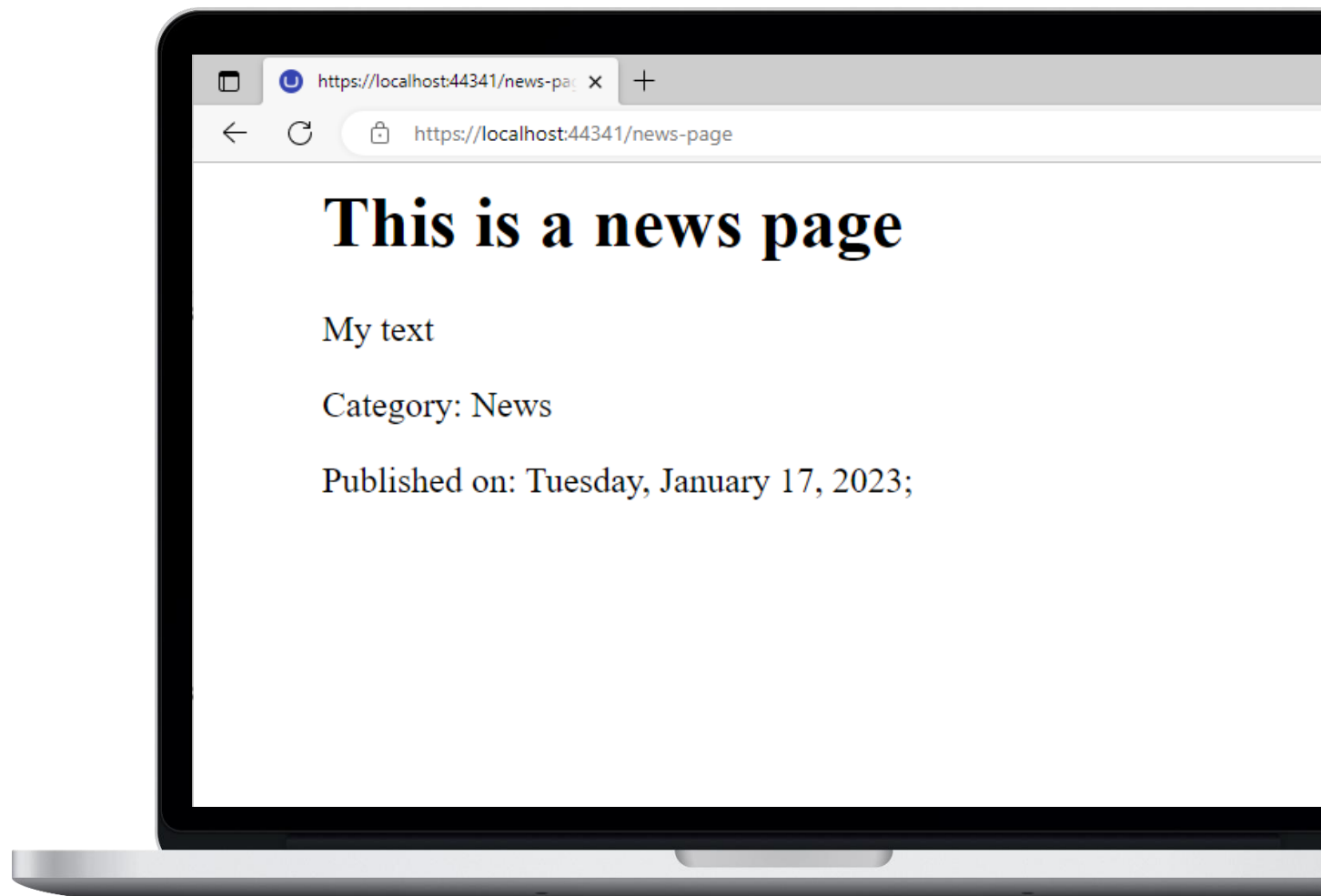
Putting it all together

Our view

```
1  @model ContentDetailPage
2
3  <h1>
4      @Model.Title
5  </h1>
6
7  <p>
8      @Model.IntroText
9  </p>
10
11 <p>
12     Category: @Model.Category
13 </p>
14
15 <p>
16     Published on: @Model.PublishDate.ToLongDateString();
17 </p>
```

Rendered page

- Rendered with Razor
- Cache content wrapper
- Decoupled from Umbraco



What does this have to do
with Headless?

Return any page as JSON

```
public override IActionResult Index()
{
    var publishedContent = UmbracoContext.PublishedRequest?.PublishedContent;

    if (publishedContent is null)
    {
        return NotFound();
    }

    var result = _contentResolver.Resolve(publishedContent);

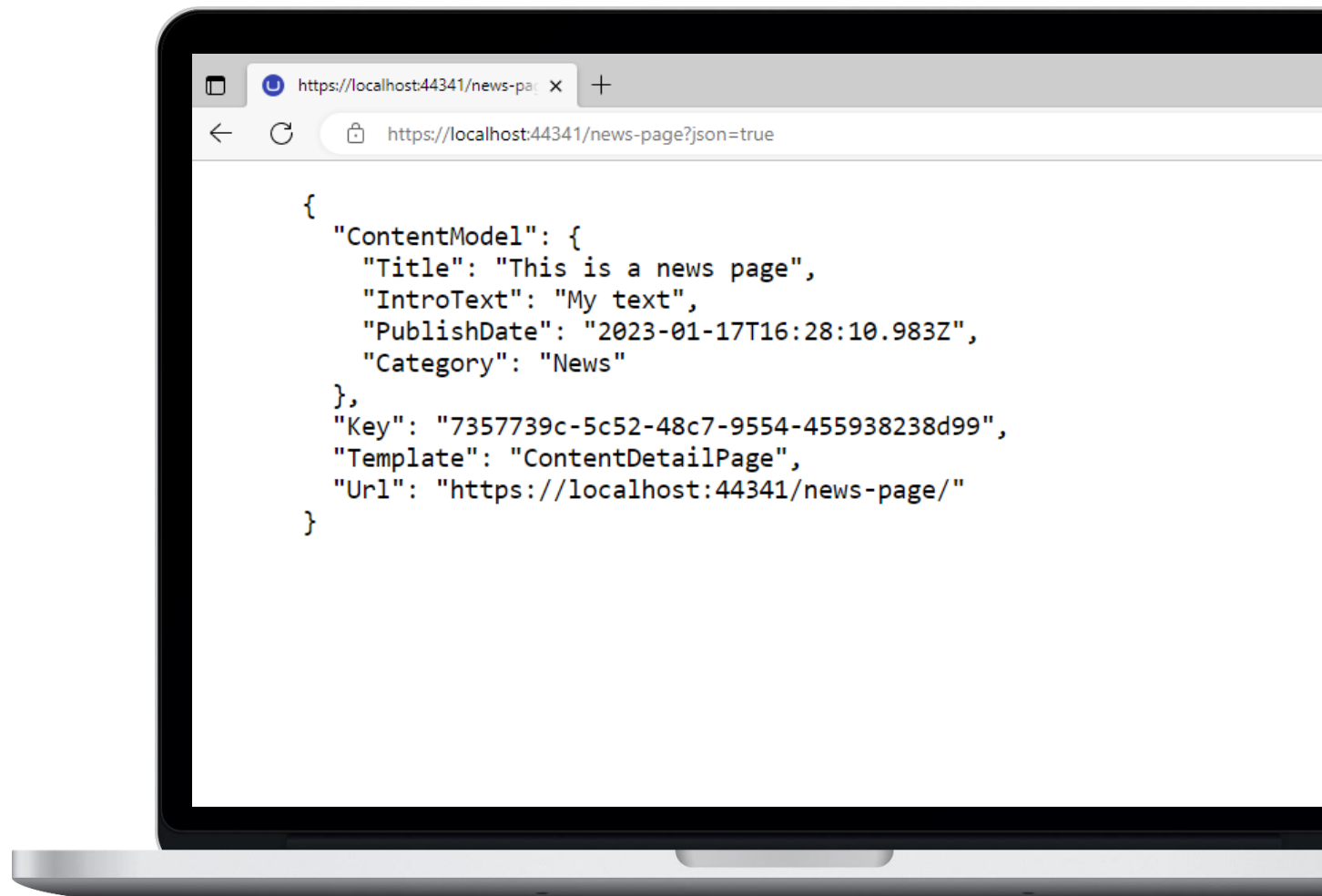
    if (Request.Query["json"] == "true")
    {
        var json = JsonSerializer.Serialize(result, new JsonSerializerOptions
        {
            WriteIndented = true,
            Converters =
            {
                new ContentModelConverter()
            }
        });

        return Ok(json);
    }

    return View(result.Template, result.ContentModel);
}
```

Infinite opportunities

Kind of “Headless”





Final thoughts

Putting it in perspective

- Looking at a prototype

Get the Source Code

berris.dev/blogs/going-headless-with-mvc/





It's a wrap

berris.dev/blogs/going-headless-with-mvc